10/652,753 PTO-892

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
16 October 2003 (16.10.2003)

PCT

(10) International Publication Number
WO 03/085557 A1

(51) International Patent Classification⁷: G06F 17/30

(21) International Application Number: PCT/IB03/01362

(22) International Filing Date: 2 April 2003 (02.04.2003)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
10/119,605 10 April 2002 (10.04.2002) US

(71) Applicant: KONINKLIJKE PHILIPS ELECTRONICS N.V. [NL/NL]; Groenewoudseweg 1, NL-5621 BA Eindhoven (NL).

(72) Inventor: HU, Jingkun; c/o Prof. Holstlaan 6, NL-5656 AA Eindhoven (NL).

(74) Agent: GROENENDAAL, Antonius, W., M.; Internationaal Octrooibureau B.V., Prof. Holstlaan 6, NL-5656 AA Eindhoven (NL).

(81) Designated States (*national*): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NI, NO, NZ, OM, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States (*regional*): ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, RO, SE, SI, SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

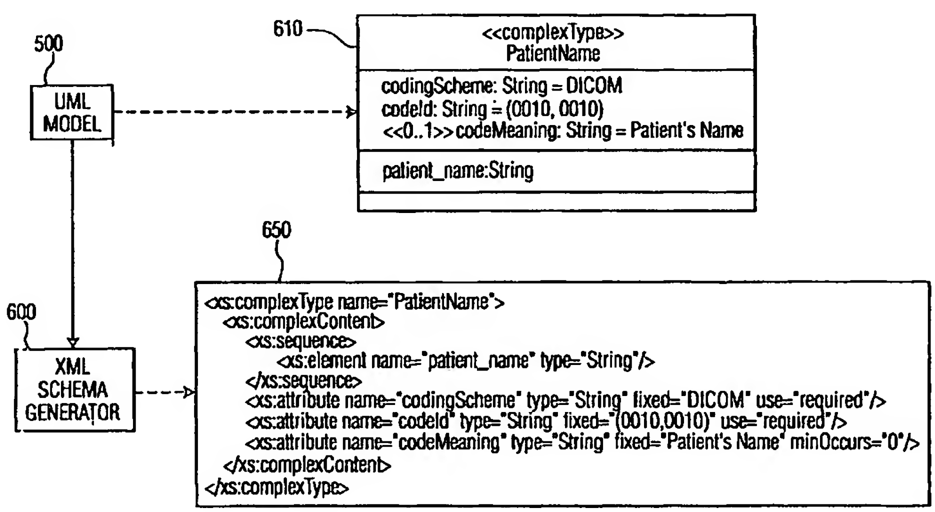
Declaration under Rule 4.17:
— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii)) for the following designation CN*

Published:
— *with international search report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

WO 03/085557 A1

(54) Title: METHOD AND APPARATUS FOR MODELING EXTENSIBLE MARKUP LANGUAGE (XML) APPLICATIONS USING THE UNIFIED MODELING LANGUAGE (UML)



(57) Abstract: A method and apparatus are disclosed for modeling XML applications using an extended UML notation. The present invention extends the UML model template to include an additional compartment, referred to as a constant attribute compartment. The constant attribute compartment allows the constant attributes of an XML element to beexplicitly represented in the UML model. The disclosed UML model can thereby distinguish XML elements and XML attributes using the additional constant attribute compartment to represent attributes that remain constant through all instances of an object. The UML modeling of XML elements is more clear and readable by separately modeling the constant attributes and the variable attributes of the XML element. The present invention facilitates the automatic generation of XML schemas and source code with the appropriate software tools.

Method and apparatus for modeling extensible markup language (XML) applications using the unified modeling language (UML)

FIELD OF THE INVENTION

The present invention relates to methods and apparatus for modeling XML applications, and more particularly, to methods and apparatus for modeling XML applications using a UML notation.

BACKGROUND OF THE INVENTION

The Extensible Markup Language (XML), recommended by the World Wide Web Consortium (W3C), is a popular standard for encoding textual information. For a discussion of the XML standard, see, for example, Extensible Markup Language (XML) 1.0 W3C Recommendation, http://www.w3.org/TR/1998/REC-xml-19980210. The XML standard allows XML-enabled applications to inter-operate with other compliant systems for the exchange of encoded information.

As XML becomes more popular with information technology professionals, the ability to accurately model XML applications is becoming increasingly important. Currently, most XML applications are modeled using the Unified Modeling Language (UML), standardized by the Object Management Group (OMG), and described in www.omg.org/uml.

UML modeling is particularly desirable since the source code for an application that is modeled in accordance with the UML standard can be automatically generated in a number of common programming languages, such as Java and C++, using available UML tools. In addition, an XML Document Type Definition (DTD) can be generated from UML models using available XML Metadata Interchange (XMI) techniques.

FIG. 1 illustrates the current UML notation for modeling objects. As shown in FIG. 1, a class element is represented in UML using a unit 100 having three compartments 110, 120, 130. A class name compartment 110 records the name of the corresponding class. An attribute compartment 120 indicates one or more attribute names and their corresponding type. An operation compartment 130 indicates one or more operation names with their corresponding signatures.

The UML standard was originally developed to model object-oriented applications. The notation shown in FIG. 1 works well for the modeling of objects in object-oriented applications, where a class has a set of attributes and operations. With the increasing use of distributed computing, however, the constant attributes of a class, such as unique class identifier, are important for identifying such a class or instance of this class. A need therefore exists for an extension of the UML model that allows the constant attributes of a class to be explicitly indicated in the UML model. A further need exists for a method and apparatus that allows an XML document or application object to be more precisely expressed.

A number of techniques have been proposed or suggested for modeling XML applications with UML or variations of UML. For a discussion of conventional techniques for modeling XML applications with UML, see, for example, David Carlson, "Modeling XML Applications with UML – Practical e-Business Applications, Addison-Wesley (2001). XML data itself does not have behaviors or operations but only attributes and elements. Generally, the basic unit of XML data is an element. An element has a value or has its own attributes (or both). In the current UML notation, however, it is difficult to differentiate between XML elements and attributes because they are typically put in the attribute compartment 120.

SUMMARY OF THE INVENTION

Generally, a method and apparatus are disclosed for modeling XML applications using an extended UML notation. The present invention extends the UML model template to include an additional compartment, referred to herein as a constant attribute compartment. The constant attribute compartment allows the constant attributes of an XML element to be explicitly represented in the UML model. In this manner, the disclosed UML model distinguishes XML elements and XML attributes using the additional constant attribute compartment to store attributes that generally remain constant through all instances of an XML element type.

The present invention makes the UML modeling of XML elements more clear and readable by separately modeling the attributes and elements of the XML element type. In addition, the present invention facilitates the automatic generation of XML schemas and source code with the appropriate software tools. An XML schema generator is disclosed that uses the extended UML model template of the present invention to automatically generate

XML schemas. A source code generator is disclosed that uses the extended UML model template of the present invention to generate source code in a given programming language.

A more complete understanding of the present invention, as well as further features and advantages of the present invention, will be obtained by reference to the following detailed description and drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 illustrates a conventional unit for representing an object in accordance with the current UML model;

FIG. 2 illustrates a first conventional approach for modeling XML applications with UML or variations of UML;

FIG. 3 illustrates a second conventional approach for modeling XML applications with UML or variations of UML;

FIG. 4 illustrates a third conventional approach for modeling XML applications with UML or variations of UML;

FIG. 5 illustrates a new UML notation in accordance with the present invention for modeling XML elements;

FIG. 6 illustrates an XML schema generator that generates XML schemas using the UML model template of the present invention;

FIG. 7 illustrates a source code generator that generates source code a given programming language using the UML model template of the present invention; and

FIG. 8 is a block diagram of a system suitable for implementing all or a portion of the present invention.

DETAILED DESCRIPTION

As discussed hereinafter, the present invention extends the UML model to allow the constant attributes of a class to be explicitly indicated in the UML model. Thus, the present invention provides methods and apparatus that allow XML documents or application objects to be more precisely expressed. In particular, the disclosed UML model distinguishes XML elements and XML attributes by inserting an additional constant attribute compartment for containing attributes that generally remain constant through all instances of an XML element type.

In UML, attributes are member/data variables of a class. These attributes can be divided into two categories: constant attributes such as class identification and variable

attributes which vary from objects to objects. Here, attributes we mention are UML attributes not XML attributes.

Modeling XML Applications With UML

A number of techniques have been proposed or suggested for modeling XML applications with UML or variations of UML. FIG. 2 illustrates one conventional approach for modeling XML applications using UML. As shown in FIG. 2, the exemplary class name compartment 210 records the name of the corresponding class or element, PatientName. A label field and an id field in the attribute compartment 220 hold attributes of the element, PatientName. The field patient_name holds the value of the element, PatientName. As shown in FIG. 2, an XML schema 240 is manually generated from the UML class model 200. The operation compartment 230 is not used for modeling of XML objects. Finally, actual instances of the schema 250 can be generated using the XML schema 240.

The problem with the first approach shown in FIG. 2 is that the model itself does not indicate which fields in the attribute compartment 220 are mapped to XML elements or attributes.

FIG. 3 illustrates another conventional approach for modeling XML applications using UML. As shown in FIG. 3, the exemplary class name compartment 310 records the name of the corresponding class or element, PatientName. A label field and an id field in the attribute compartment 320 hold attributes of the element, PatientName. The field patient_name holds the value of the element, PatientName. Unlike the first approach in FIG. 2, the label, id and patient_name fields are marked as being an attribute or an element. As shown in FIG. 3, an XML schema 340 is manually generated from the UML class model 300. The operation compartment 330 is not used for modeling of XML objects. Finally, actual instances of the schema 350 can be generated using the XML schema 340.

The problem with the second approach shown in FIG. 3 is that the model is too verbose and the mapping rules must be applied to each UML attribute to evaluate its target. The UML attribute usage cannot be specified.

FIG. 4 illustrates yet another conventional approach for modeling XML applications using UML, where two UML units are used to represent each XML element. A first unit 400-1 holds the variable attribute information about the element and a second unit 400-2 holds the constant attribute information about the element. As shown in FIG. 4, the exemplary class name compartment 410-1 of a first unit 400-1 records the name of the corresponding class or element, PatientName. The field patient_name in the attribute compartment 420-1 holds the value of the element, PatientName.

A label field and an id field in the attribute compartment 420-2 of the second unit 400-2 hold the constant attributes of the element, PatientName. As shown in FIG. 4, an XML schema 440 can be automatically generated from the UML class model 400-1, 2. The operation compartment 430 is not used for modeling of XML objects. Actual instances of the schema (not shown) can be generated using the XML schema 440.

The problem with the third approach shown in FIG. 4 is that the model requires a separate, additional class to define the attributes.

Extended UML for Modeling XML Applications

FIG. 5 illustrates a new UML notation in accordance with the present invention for modeling XML elements. As shown in FIG. 5, an XML element is represented using a UML unit 500 having four compartments 510, 520, 530, 540. A class name compartment 510 records the name of the corresponding class or XML element type. The class name compartment 510 allows a stereotype to be specified, such as simpleType and complexType in XML structures (see, for example, XML Schema Definition Language, www.w3c.org/XML/Schema).

A constant attribute compartment 520 specifies the constant attributes that will remain constant through all the PatientName instances. A variable attribute compartment 530 specifies the variable attributes that will vary for different PatientName instances. The operation compartment 540 indicates one or more operation names with their corresponding return type and parameters, and is typically empty in defining XML types. In compartment 520, <<0..1>> specifies the usage of the attribute label as optional. Similarly, the usage of other attributes can also be specified. The numbers within the brackets "<< >>" specify the minimum and maximum occurrences of a given attributes.

Applications of Extended UML Model

Once a UML model is built using the UML model template 500 in accordance with the present invention, XML schemas and source code in a given programming language can be automatically generated using existing tools together with additional mapping rules as illustrated in FIGS. 6 and 7.

FIG. 6 illustrates an XML schema generator 600 that generates XML schemas 650 using the UML model template 500. The XML schema generator 600 may be embodied, for example, as a computer or workstation that employs the XML Metadata Interchange (XMI) toolkit, commercially available from IBM Corp. and described in http://www.alphaworks.ibm.com/tech/xmiframework, or similar tools, such as the HyperModel™ application commercially available from XMLModeling Corp. and described

in www.xmlmodeling.com, as modified herein to incorporate the modified notation/template of the present invention, as would be apparent to a person of ordinary skill in the art.

In one exemplary embodiment, the XML schema generator 600 maps a UML class to an XML Schema component, such as complexType, element, or simpleType, based on its stereotype. The constant attributes are mapped to the attributes of this component. Likewise, the variable attributes are mapped to the elements of this component.

FIG. 7 illustrates a source code generator 700 that generates Java source code 750-1, 750-2, using the UML model template 500. The source code generator 700 may be embodied, for example, as a computer or workstation that employs the Rational Rose™ tool, commercially available from Rational Software Corporation and described in http://www.rational.com, or similar tools, such as System Architect™ from Popkin Software Corporation, and described in www.popkin.com, as modified herein to incorporate the modified notation/template of the present invention, as would be apparent to a person of ordinary skill in the art.

Referring now to FIG. 8, a block diagram is shown of an exemplary system 800 suitable for carrying out embodiments of the present invention. System 800 could be used for some or all of the methods and systems disclosed in FIGS. 5 through 7. System 800 comprises a computer system 810 and a Compact Disk (CD) 850. Computer system 810 comprises a processor 820, a memory 830 and a video display 840.

As is known in the art, the methods and apparatus discussed herein may be distributed as an article of manufacture that itself comprises a computer-readable medium having computer-readable code means embodied thereon. The computer-readable program code means is operable, in conjunction with a computer system such as computer system 810, to carry out all or some of the steps to perform the methods or create the apparatuses discussed herein. The computer-readable medium may be a recordable medium (e.g., floppy disks, hard drives, compact disks, or memory cards) or may be a transmission medium (e.g., a network comprising fiber-optics, the world-wide web, cables, or a wireless channel using time-division multiple access, code-division multiple access, or other radio-frequency channel). Any medium known or developed that can store information suitable for use with a computer system may be used. The computer-readable code means is any mechanism for allowing a computer to read instructions and data, such as magnetic variations on a magnetic medium or height variations on the surface of a compact disk, such as compact disk 850.

Memory 830 configures the processor 820 to implement the methods, steps, and functions disclosed herein. The memory 830 could be distributed or local and the

processor 820 could be distributed or singular. The memory 830 could be implemented as an electrical, magnetic or optical memory, or any combination of these or other types of storage devices. Moreover, the term "memory" should be construed broadly enough to encompass any information able to be read from or written to an address in the addressable space accessed by processor 810. With this definition, information on a network is still within memory 830 because the processor 820 can retrieve the information from the network. It should be noted that each distributed processor that makes up processor 820 generally contains its own addressable memory space. It should also be noted that some or all of computer system 810 can be incorporated into an application-specific or general-use integrated circuit.

Video display 840 is any type of video display suitable for interacting with a human user of system 800. Generally, video display 840 is a computer monitor or other similar video display.

It is to be understood that the embodiments and variations shown and described herein are merely illustrative of the principles of this invention and that various modifications may be implemented by those skilled in the art without departing from the scope and spirit of the invention.

CLAIMS:

1. A method for representing an XML object using a UML model, comprising the steps of:

providing a UML model unit (500) for representing said XML object, said UML model unit (500) having a plurality of compartments (510, 520, 530, 540); and

providing separate compartments (520, 530) in said UML model unit (500) for representing constant and variable attributes of said XML object.

2. The method of claim 1, wherein said constant attribute compartment (520) stores attributes that remain constant through all instances of an object.

3. The method of claim 1, wherein said variable attribute compartment stores attributes that vary for different instances of an object.

4. The method of claim 1, wherein said constant attributes are automatically mapped to attributes of an XML schema (650) component.

5. The method of claim 1, wherein said variable attributes are automatically mapped to elements of an XML schema (650) component.

6. A method for generating an XML schema (650) from a UML model, comprising the steps of:

modeling said XML schema (650) using an extended UML model unit (500) having separate compartments (520, 530) for representing constant and variable attributes of said XML object; and

automatically generating said XML schema (650) from said modeled XML schema.

7. The method of claim 6, wherein said step of automatically generating said XML schema (650) further comprises the step of mapping constant attributes to attributes of an XML schema (650) component.

8. The method of claim 6, wherein said step of automatically generating said XML schema (650) further comprises the step of mapping variable attributes to elements of an XML schema (650) component.

9. A method for generating source code (750) from a UML model, comprising the steps of:

modeling said source code (750) using an extended UML model unit (500) having separate compartments (520, 530) for representing constant and variable attributes of said XML object; and

automatically generating said source code (750) from said modeled XML schema (650).

10. The method of claim 9, wherein said step of automatically generating said source code (750) further comprises the step of mapping constant attributes to constant member variables of a class.

11. The method of claim 9, wherein said step of automatically generating said source code (750) further comprises the step of mapping variable attributes to member variables other than constant variables of a class.

12. A system (800) for representing an XML object using a UML model, comprising:

a memory (830) for storing computer readable code; and

a processor (820) operatively coupled to said memory (830), said processor (820) configured to:

provide a UML model unit (500) for representing said XML object, said UML model unit (500) having a plurality of compartments (510, 520, 530, 540); and

provide separate compartments (520, 530) in said UML model unit (500) for representing constant and variable attributes of said XML object.

13. A system (800) for generating an XML schema (650) from a UML model, comprising:

a memory (830) for storing computer readable code; and

a processor (820) operatively coupled to said memory (830), said processor (820) configured to:

model said XML schema (650) using an extended UML model unit (500) having separate compartments (520, 530) for representing constant and variable attributes of said XML object; and

automatically generate said XML schema (650) from said modeled XML schema (650).

14. A system (800) for generating source code (750) from a UML model, comprising:

a memory (830) for storing computer readable code; and

a processor (820) operatively coupled to said memory (830), said processor (820) configured to:

model said XML schema (650) using an extended UML model unit (500) having separate compartments (520, 530) for representing constant and variable attributes of said XML object; and

automatically generate said source code (750) from said modeled XML schema (650).

15. An article of manufacture (850) for representing an XML object using a UML model, comprising:

a computer readable medium having computer readable code means embodied thereon, said computer readable program code means comprising:

a step to provide a UML model unit (500) for representing said XML object, said UML model unit (500) having a plurality of compartments (510, 520, 530, 540); and

a step to provide separate compartments (520, 530) in said UML model unit (500) for representing constant and variable attributes of said XML object.

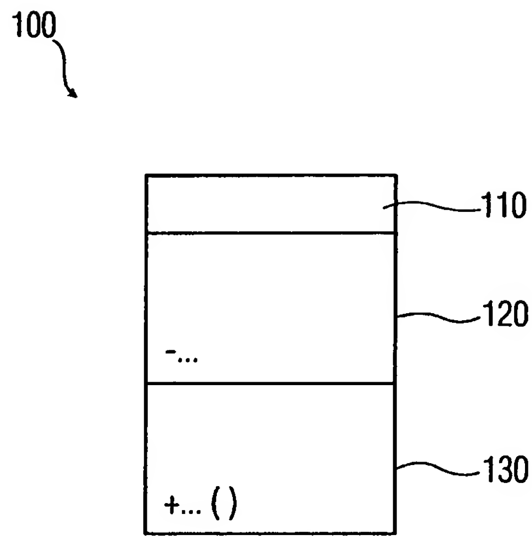


FIG. 1

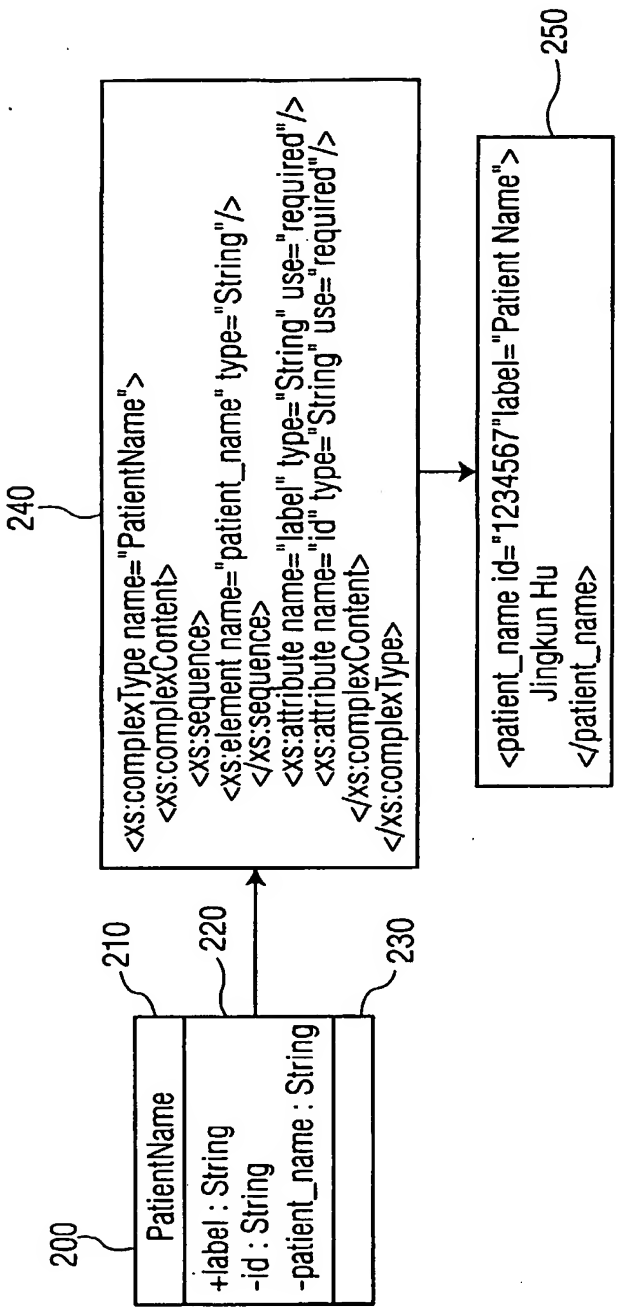

200
PatientName
+label : String
-id : String
-patient_name : String
210
220
230
240
<xs:complexType name="PatientName">
<xs:complexContent>
<xs:sequence>
<xs:element name="patient_name" type="String"/>
</xs:sequence>
<xs:attribute name="label" type="String" use="required"/>
<xs:attribute name="id" type="String" use="required"/>
</xs:complexContent>
</xs:complexType>
250
<patient_name id="1234567"label="Patient Name">
Jingkun Hu
</patient_name>

FIG. 2

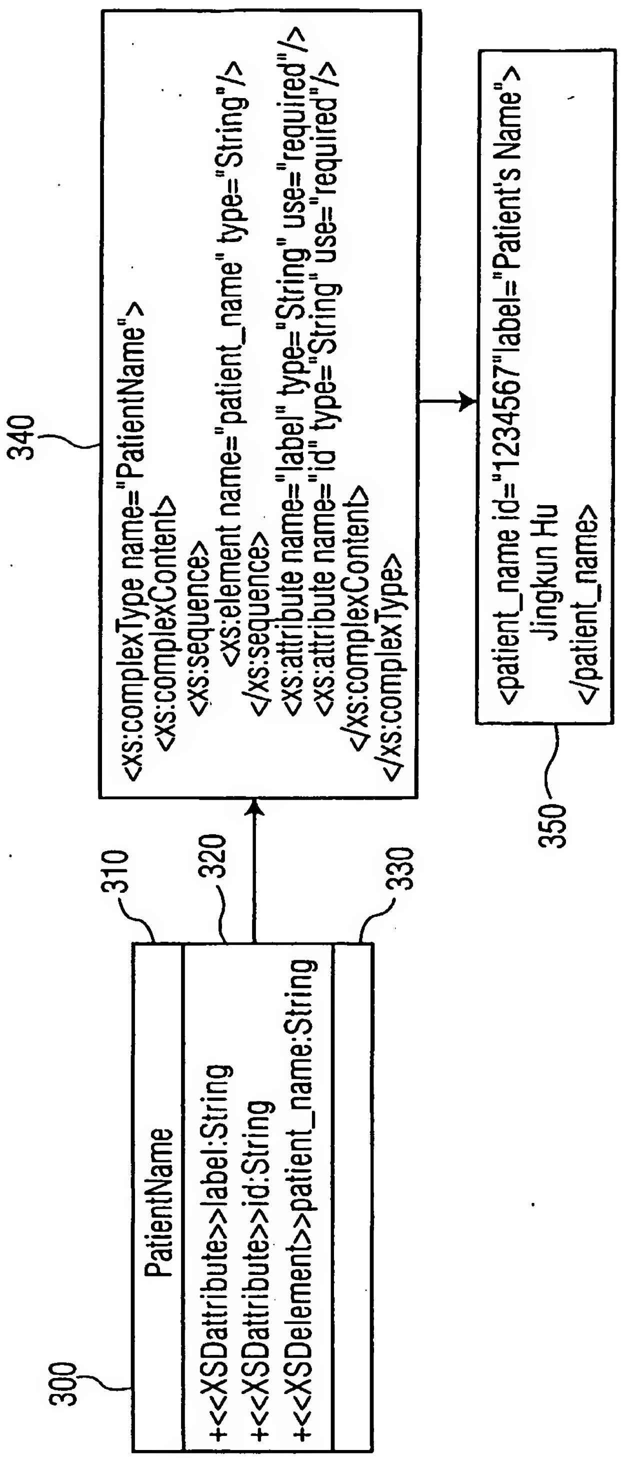

300
PatientName
+<<XSDattribute>>label:String
+<<XSDattribute>>id:String
+<<XSDelement>>patient_name:String
310
320
330
340
<xs:complexType name="PatientName">
<xs:complexContent>
<xs:sequence>
<xs:element name="patient_name" type="String"/>
</xs:sequence>
<xs:attribute name="label" type="String" use="required"/>
<xs:attribute name="id" type="String" use="required"/>
</xs:complexContent>
</xs:complexType>
<patient_name id="1234567"label="Patient's Name">
Jingkun Hu
</patient_name>
350

FIG. 3

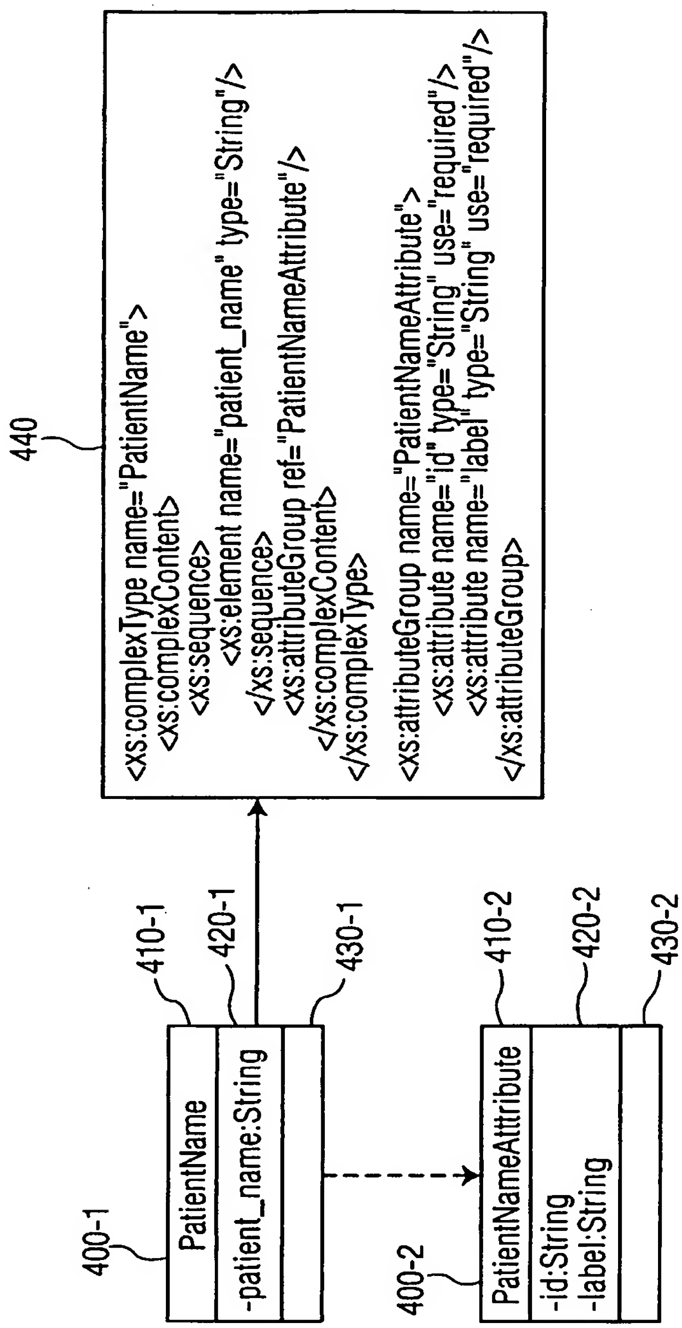

440
<xs:complexType name="PatientName">
<xs:complexContent>
<xs:sequence>
<xs:element name="patient_name" type="String"/>
</xs:sequence>
<xs:attributeGroup ref="PatientNameAttribute"/>
</xs:complexContent>
</xs:complexType>
<xs:attributeGroup name="PatientNameAttribute">
<xs:attribute name="id" type="String" use="required"/>
<xs:attribute name="label" type="String" use="required"/>
</xs:attributeGroup>
400-1
410-1
420-1
430-1
PatientName
-patient_name:String
400-2
410-2
420-2
430-2
PatientNameAttribute
-id:String
-label:String

FIG. 4

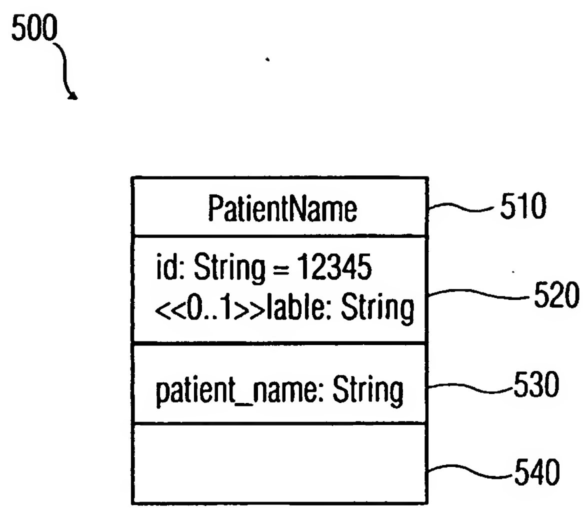

500
PatientName
510
id: String = 12345
<<0..1>>lable: String
520
patient_name: String
530
540

FIG. 5

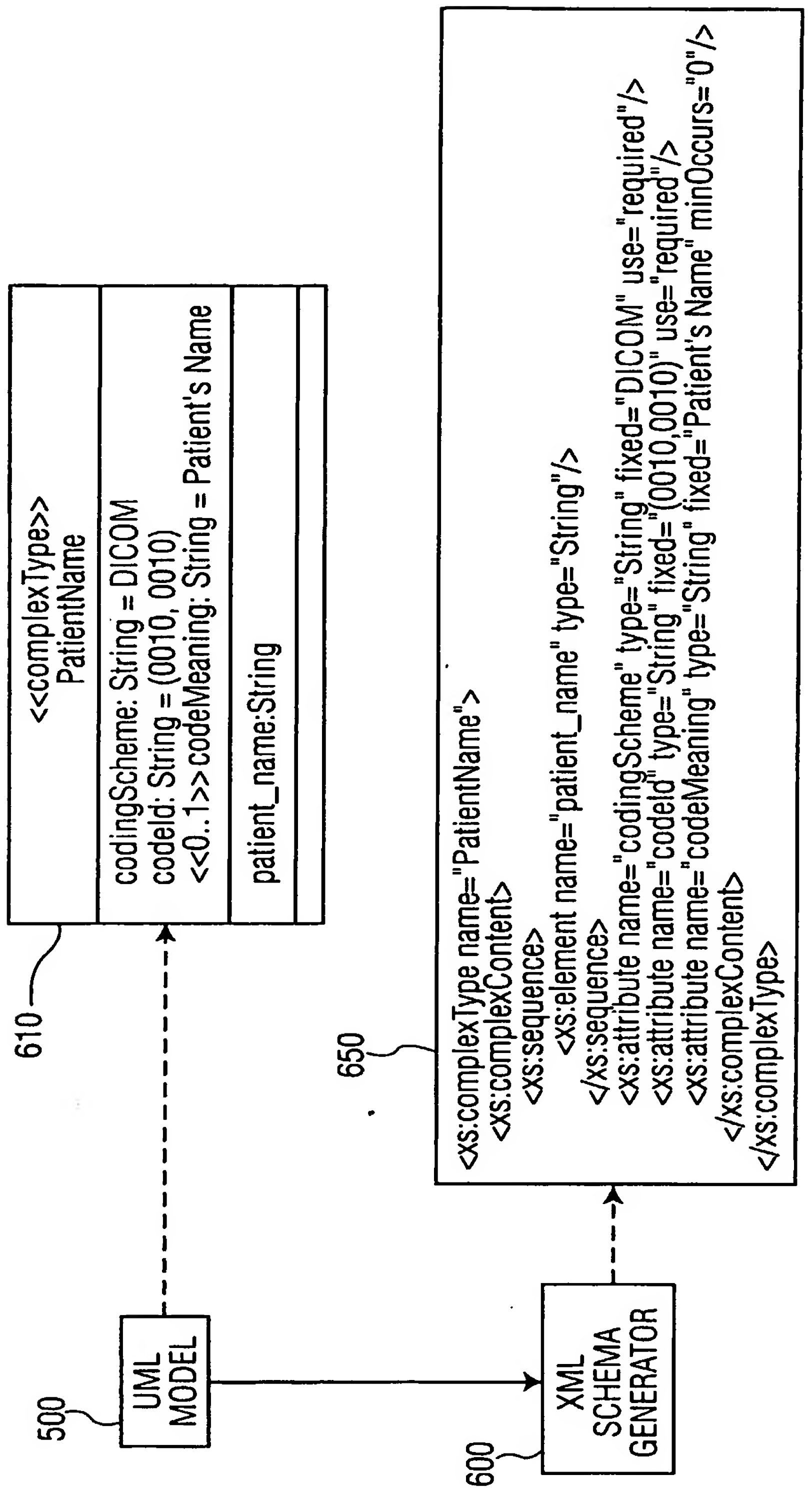

610
<<complexType>>
PatientName
codingScheme: String = DICOM
codeId: String = (0010, 0010)
<<0..1>> codeMeaning: String = Patient's Name
patient_name:String
650
<xs:complexType name="PatientName">
<xs:complexContent>
<xs:sequence>
<xs:element name="patient_name" type="String"/>
</xs:sequence>
<xs:attribute name="codingScheme" type="String" fixed="DICOM" use="required"/>
<xs:attribute name="codeId" type="String" fixed="(0010,0010)" use="required"/>
<xs:attribute name="codeMeaning" type="String" fixed="Patient's Name" minOccurs="0"/>
</xs:complexContent>
</xs:complexType>
500
UML MODEL
600
XML SCHEMA GENERATOR

FIG. 6

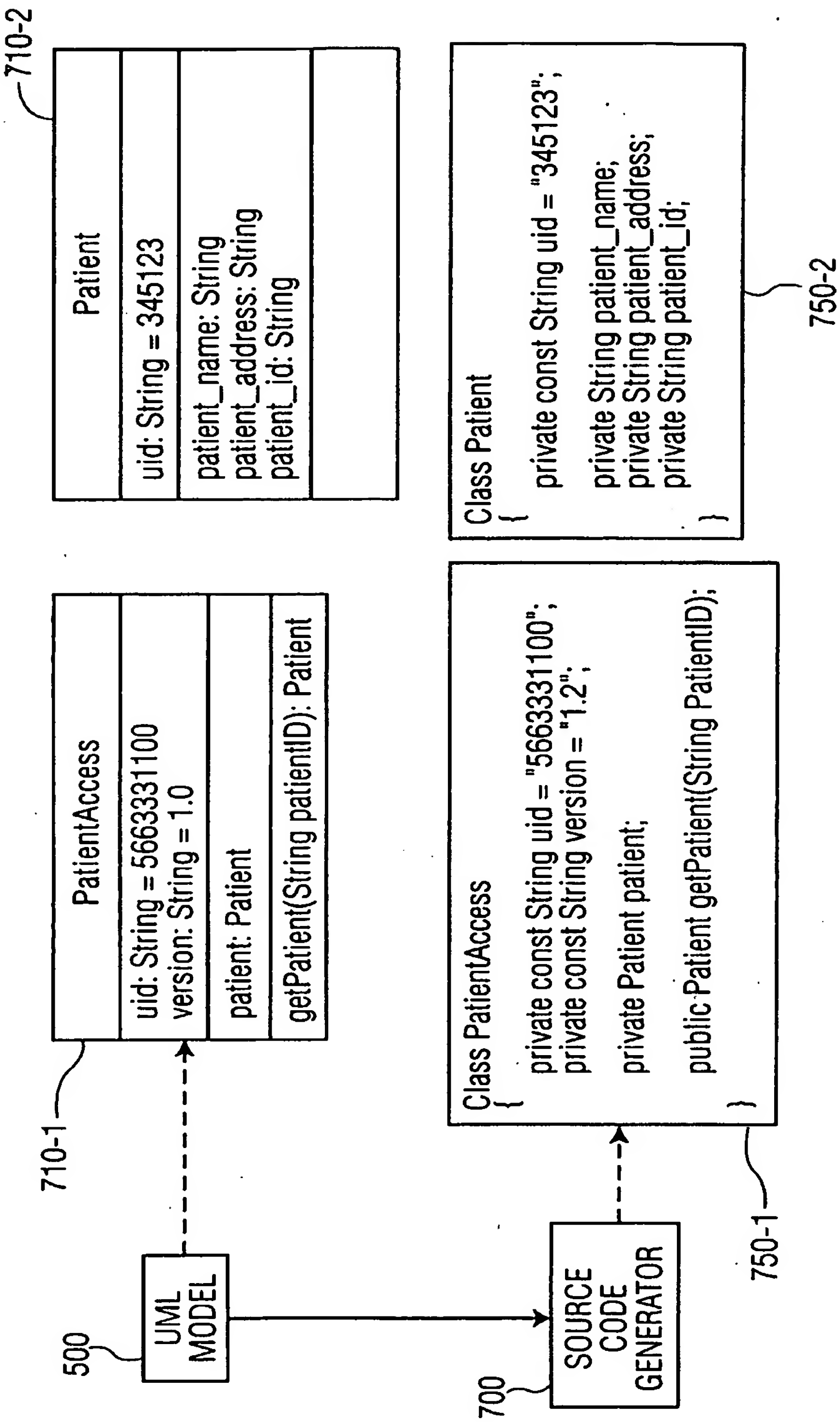

710-2
Patient
uid: String = 345123
patient_name: String
patient_address: String
patient_id: String
710-1
PatientAccess
uid: String = 5663331100
version: String = 1.0
patient: Patient
getPatient(String patientID): Patient
500
UML MODEL
700
SOURCE CODE GENERATOR
Class Patient
{
private const String uid = "345123";
private String patient_name;
private String patient_address;
private String patient_id;
}
750-2
Class PatientAccess
{
private const String uid = "5663331100";
private const String version = "1.2";
private Patient patient;
public Patient getPatient(String PatientID);
}
750-1

FIG. 7

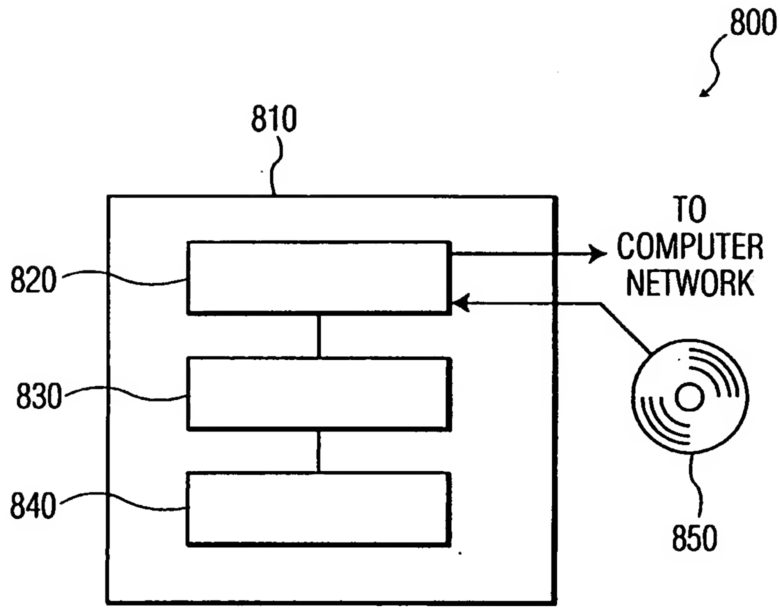


FIG. 8

INTERNATIONAL SEARCH REPORT

Internatiol ɔlication No
PCT/IB 03/01362

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G06F17/30

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

EPO-Internal, WPI Data, PAJ, INSPEC, COMPENDEX, IBM-TDB

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	JENSEN M R ET AL: "Converting XML Data to UML Diagrams for Conceptual Data Integration " DATA INTEGRATION OVER THE WEB, FIRST INTERNATIONAL WORKSHOP, INTERLAKEN, SWITZERLAND, INFORMAL PROCEEDINGS, 'Online! 4 June 2001 (2001-06-04), XP002246209 Retrieved from the Internet: <URL:http://www.cs.auc.dk/{mrj/publications/diweb.pdf> 'retrieved on 2003-07-02!	1-5,12, 15
A	the whole document	9-11,14
	-/--	

[X] Further documents are listed in the continuation of box C.

[ ] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
2 July 2003	17/07/2003

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Stauch, M

Form PCT/ISA/210 (second sheet) (July 1992)

Internatio ›plication No
PCT/IB 03/01362

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
X	CARLSON D: "Modeling XML Vocabularies with UML: Part III" XMLMODELLING.COM, 'Online! 28 September 2001 (2001-09-28), XP002246210 Retrieved from the Internet: <URL:http://xmlmodeling.com/articles/ModelingXML-Part3.pdf> 'retrieved on 2003-07-02!	6-8,13
A	the whole document	9-11,14
X	GRAU A ET AL: "The TROLL approach to conceptual modelling: syntax, semantics and tools" CONCEPTUAL MODELING - ER'98. 17TH INTERNATIONAL CONFERENCE ON CONCEPTUAL MODELING. PROCEEDINGS, CONCEPTUAL MODELING - ER'98. 17TH INTERNATIONAL CONFERENCE ON CONCEPTUAL MODELING. PROCEEDINGS, SINGAPORE, 16-19 NOV. 1998, pages 277-290, XP002246211 1998, Berlin, Germany, Springer-Verlag, Germany ISBN: 3-540-65189-6	9-11,14
A	page 5, line 8 -page 6, line 6; figures 1,2 page 12, line 25 - line 31	1-8,12, 13,15
X	ANONYMOUS: "Mapping Attributes (UML Suite Code Generation)" ANONYMOUS, 'Online! XP002246212 Retrieved from the Internet: <URL:http://www.ifi.uio.no/in219/verktoy/doc/html/doc/code_gens/pb/pbgen2.html> 'retrieved on 2003-07-02!	9-11,14
A	page 3, line 17 -page 4, line 8	1-8,12, 13,15